Stegräknare

Programmering av inbyggda system
Mjukvaruutvecklare inbyggda system 2016

EJ FÄRDIG RAPPORT

Dennis Bunne
Simon Karlsson
Johan Kämpe

Sammanfattning

Skrivs när rapport är klar.

Innehållsförteckning

1 Inledning

1.1 Syfte

Syftet med projektet är att tillverka en bärbar, batteridriven stegräknare. Stegräknaren ska vara modulär. Det vill säga, att det ska gå att ansluta och ta bort enheter, display, WiFi-modul m.m.

Initialt används en display för att visa antal steg. Stegen ska avläsas med hjälp av en accelerometer, eller alternativt med en vibrations sensor.

Stegräknaren kan senare utökas med hastighetsmätare, tidräkning, ESP-modul eller bluetooth-modul för att skicka data m.m, om funktionen med stegräkning fungerar korrekt.

1.2 Bakgrund

1.2.1 Stegräknare

En stegräknare är en, vanligtvis portabel, enhet som mäter hur många steg en människa tar. Den kan användas vid exempelvis löpning eller promenad. Eller för att se hur många steg en person tar under en dag. Stegräknare kan också kallas pedometer.

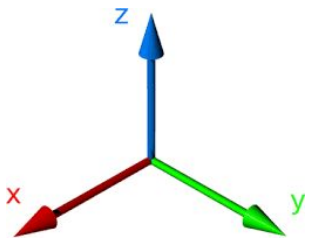
1.2.2 I²C

I²C, Inter-Integrated Circuit, är ett sätt att koppla enheter till moderkort, inbyggda system, mobiltelefoner eller andra enheter. Kommunikation sker via två ledare, SDA (Serial Data Line, datasignal) och SCL (Serial Clock, klocksignal). Flera I²C-enheter kan anslutas till samma SDA- och SCL-ledare.

1.2.3 X-Y-Z

X, Y och Z används inom projektet för att beskriva riktningar i en tredimensionell miljö. MPU-9250-enheten har sensorer för acceleration i dessa tre riktningar. När enheten förs åt ett visst håll mäts rörelsens acceleration i denna riktning.

X- Y- och Z-riktningar.



2 Genomförande och resultat

2.1 Använd programvara

Vid utveckling av stegräknaren har följande programvara använts

Program	Beskrivning
Atmel Studio 7.0	IDE för att skriva kod och kompilera till Atmel MCU.
Git	Versionshanteringsverktyg
GitKraken	Grafiskt gränssnitt för Git
Fritzing	Program för att skapa kretsscheman
Microsoft Word	Dokumentering
AVR Dudess	Program för att ladda upp koden till MCU i form av en HEX-fil.
Autodesk Fusion 360	CAD-program för att skapa ett hölje för produkten

2.2 Komponentlista

För att bygga stegräknaren har följande hårdvara använts:

Artikel	Beskrivning
Atmel ATMega328-PU DIP-28N 8-bit MCU	Microprocessor.
SparkFun MPU-9250	Enhet som mäter acceleration i X, Y och Z-riktning. Samt gyroskop, temperatur och kompass-funktionalitet.
Micro OLED Display SSD1306 128x64	Display med upplösningen 128x64.
AVR pocket programmer	Enhet för att programmera Atmega-MCU
DIL-hållare 28-pin 0.3"	Sockel för Atmega-MCU
Resistor 4,7kΩ, 2 st	Pull up-resistorer för I²C-kanaler.
Kristalloscillator 16 MHz	Kopplas till MCU för att öka klockfrekvensen från 1 MHz till 16 MHz
Kondensator 22pF, 2 st	Ger jämn ström till Oscillatorn
Litiumbatteri 18650	Strömförsörjning
Batterihållare för litiumbatteri 18650	
Grovekontakter 4-stift	För modularisering av I²C-komponenter

2.3 Utveckling av stegräknaren

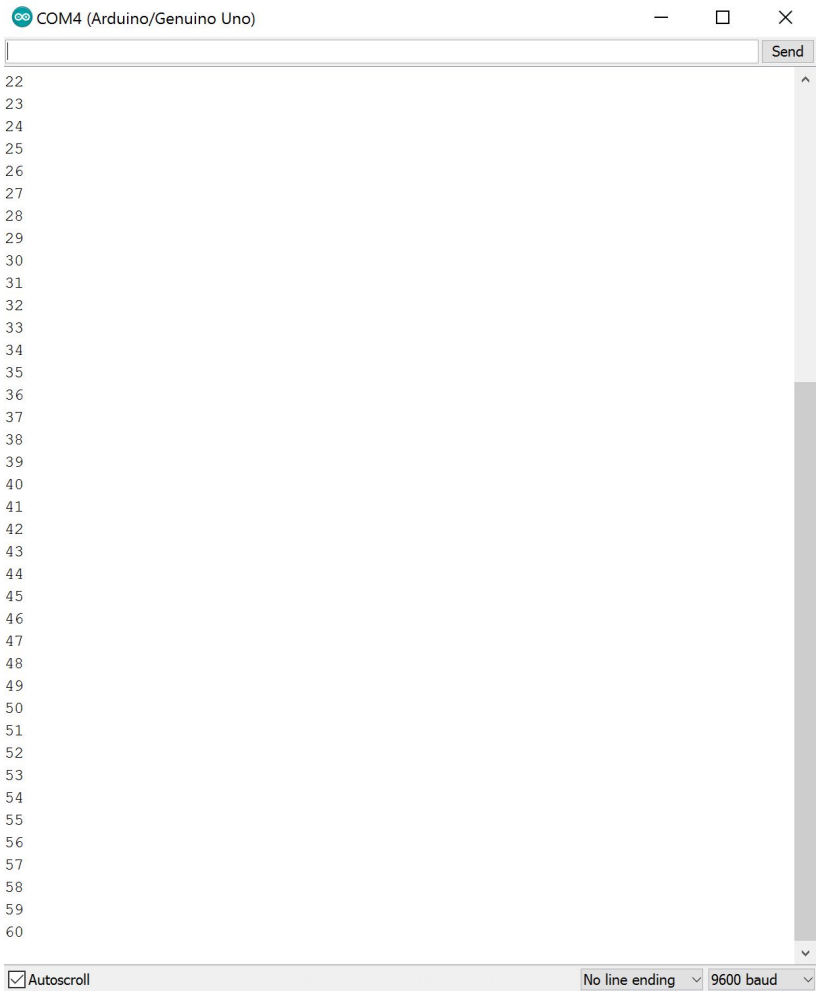
Två prototyper har byggts under projektets gång. Dessa beskrivs nedan.

2.3.1 Prototyp 1

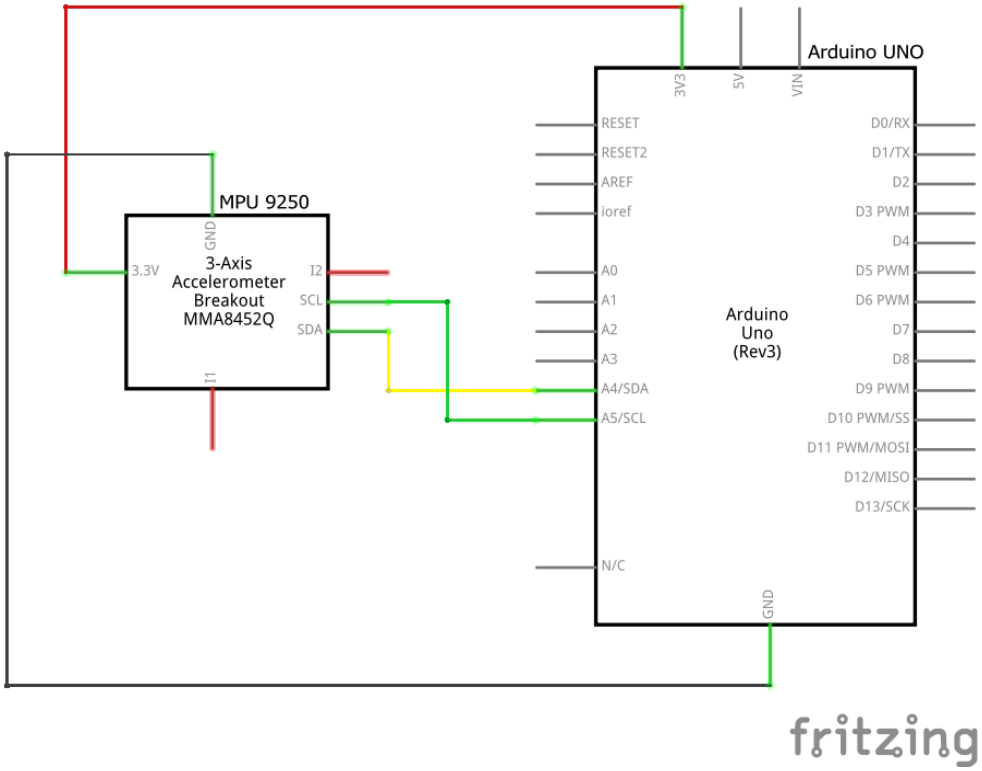
Den första prototypen byggdes med en Arduino UNO och en breadboard dit komponenter kopplades. Prototypen testades genom att stoppa ner den i en ficka och promenerades runt med. Antal steg skrevs ut till *Serial Monitor* som finns i Arduinos IDE. En bärbar dator användes för strömförsörjning via USB, och för att se datan från Serial Monitor.

Både en accelerometer (MPU-9250) och vibrationssensor testades. Accelerometern valdes att gå vidare med, då denna ansågs ge bäst och stabilast värden.

Utskrift av steg till serial monitor:



Kopplingschema för prototyp 1:



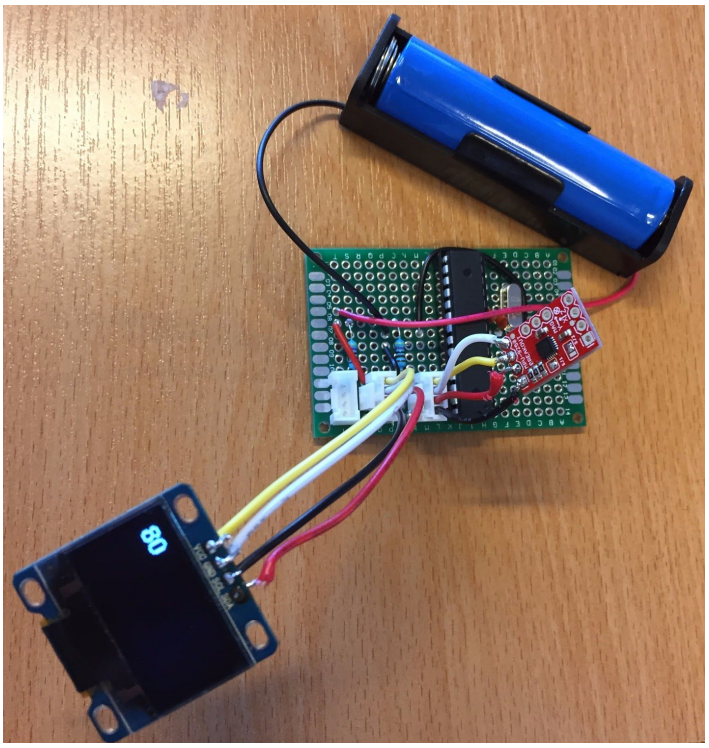
2.3.2 Prototyp 2

Till prototyp nummer 2 används en Atmega328p MCU (Micro controller unit) i stället för Arduino. Komponenter har lötts samman på en platta och satts i ett hölje byggt av en optisk mus.

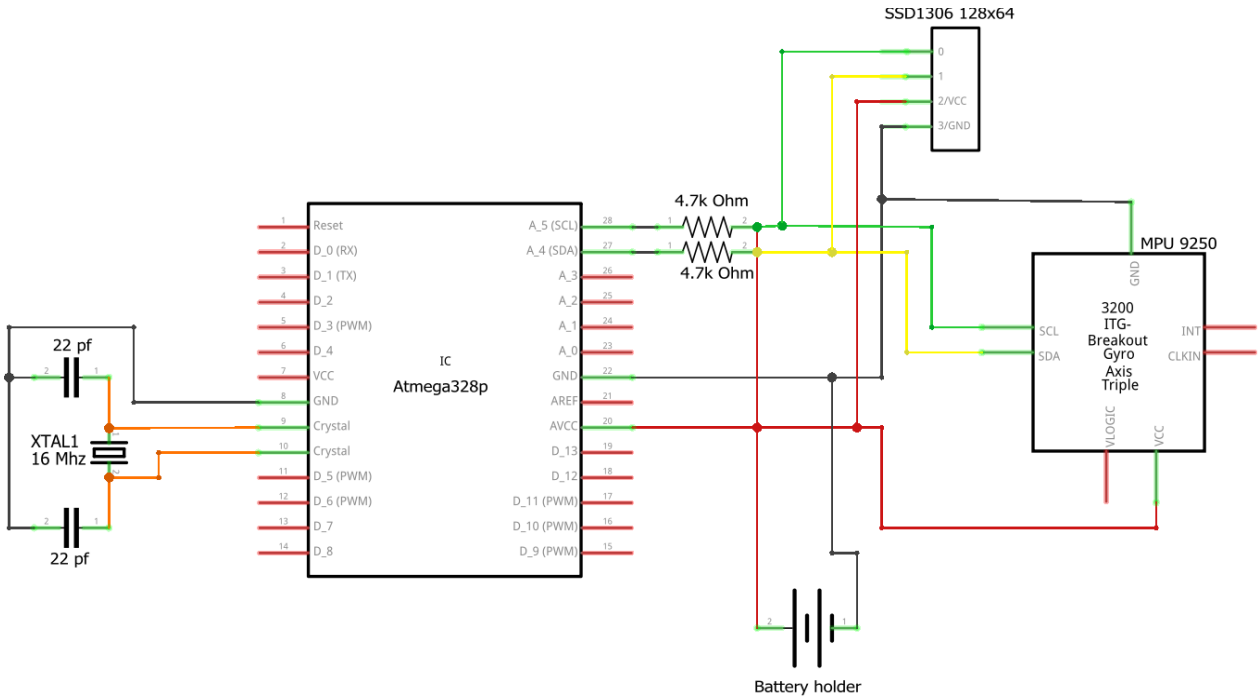
I prototyp 2 så blir vi tvungna att använda pull up resistor från SDA/SCL kontakterna. Pull up resistor är en resistor som är kopplad från kontakten(pin) till VCC(+). Det gör så att modulerna drar lite mer ström när de ska aktiveras, fast motverkar logiska fel.

Kontakter har använts för modularisering. Samtliga I2C-enheter går att byta ut eller placera om. Ett batteri används för strömförsörjning. Stegräknaren har minskat i storlek avsevärt sedan prototyp 1.

Fotografier på prototyp 2, utan och med hölje

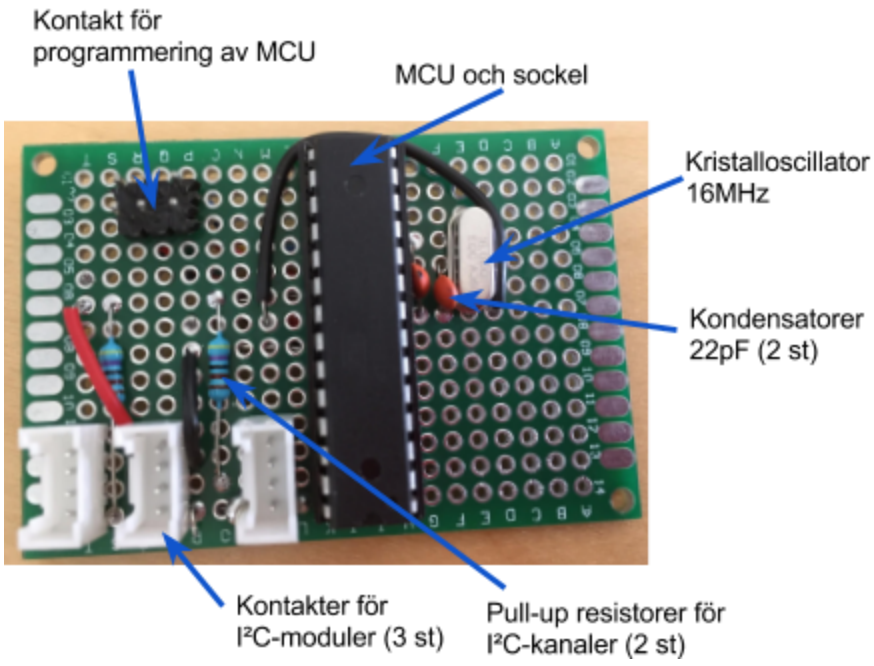


Kopplingsschema för prototyp 2



Efter sammanställning av prototyp 1, anslöts programmeringspinnar på basplattan, för programmering av mikroprocessorn via *AVR pocket programmer.*

Placering av komponenter på kopplingsplatta:



2.4 Stegräknarens funktion

Stegräknaren fungerar genom att hämta accelerationsdata från MPU-9250-enheten.

Värden för acceleration i riktningarna X, Y och Z adderas till ett enda värde. Detta sammansatta värde jämförs sedan mot ett värde som anses vara viloläge för accelerometern, dvs när den ligger helt stilla. Om skillnaden mellan dessa två värden överskrider ett visst satt tröskelvärde, så anses det att ett steg har tagits.

Vid varje taget steg uppdateras stegräknarens display med det totala antalet tagna steg.

Stegräknarens display sätts automatiskt i sömnläge om inget steg har tagits på ca 30 sekunder. Displayen vaknar automatiskt när ett steg tas.

2.5 Kod

Koden för stegräknaren är skrivet i programspråket C. Redan färdiga kodbibliotek för displayen, accelerometern och I2C har använts för att underlätta programmeringsarbetet.

Utförligare kommentering av koden finns i källkodsfilen.

2.5.1 Använda bibliotek

De bibliotek vi använde oss av var dessa :

Namn	Beskrivning
#include <stdlib.h>	Standardbibliotek som innehåller flera macros och standard funktioner för att utföra vanliga kommandon.
#include <avr/interrupt.h>	Interrupt kommer med ett par redan färdiga stopp rutiner som kommer att användas, ifall man inte specificerar sina egna. Interrupts kan vara en jobbig procedur att skapa då olika compilers använder sig utav olika stopp funktioner.
#include <avr/pgmspace.h>	Pgmspace gör så att du kan nå datan som är sparat på enheternas flash minne.
#include <math.h>	Vi använder oss utav math för att kunna använda fabs() vilket ger oss ett absolut värde.
#include "i2chw/i2cmaster.h"	***I2C master library using hardware TWI interface*** I2C-funktionalitet för AVR-enheter (Atmel MCUer) av Peter Fleury Ett färdigt bibliotek som vi hittat, skapat av Peter Fleury. Där han har satt upp interfacen för TWI (twin wire interface) eftersom detta var en väldigt stor och komplicerad bit. Har detta varit till väldigt stor nytta för oss.
#include "mpu6050/mpu6050.h"	***MPU6050 lib 0x02*** För accelerometer-enheten, via I2C. Av Davide Gironi Bibliotek för MPUn (accelerometern), där vi har skärt ner på stora bitar av koden samt gjort egna funktioner som hämtar ut data för vinklarna x,y,z.
#include "u8g/u8g.h"	**Universal 8bit Graphics Library** För display, via I2C av olikraus@gmail.com Ett färdigt bibliotek för OLED displayen.

2.5.2 Globala variabler

Följande globala variabler används i projektets kod.

Namn	Beskrivning
uint16_t steps	Håller antal steg.
double accIdle	Värde som sätts vid programmets uppstart, håller det värde som accelerometern ger när den ligger helt stilla.
double accCombined	Sammansatt accelerationsvärde från X- Y- och Z-riktning.
double accX, accY, accZ	Värden för acceleration i X- Y- och Z-riktning.
int displaySleeping	Håller displayens sömnläge, sätts till 1 om displayen är i sömnläge, annars 0.
int displaySleepTimer	Används som timer för displayens sömnläge. Ökar med 1 för varje varv i programmets huvudloop.
u8g_t u8g	Struct för biblioteket ***Universal 8bit Graphics Library****, för display.*

2.5.3 Definitioner / macron

Följande macron används i projektets kod

Namn	Värde	Beskrivning
X	**0x3B**	Address för accelerometerdata i X-riktning i MPU
Y	**0x3D**	Address för accelerometerdata i Y-riktning i MPU
Z	**0x3F**	Address för accelerometerdata i Z-riktning i MPU
MPU6050_ADDR	**(0x68 <<1)**	I²C-address för MPU-9250. Satt i *mpu6050.h*
MPU6050_AGAIN	**16384.0 KANSKE**	Värde för att omvandla rå accelerationsdata från MPU-9250 till g-krafter.
I2C_WRITE	**0**	Datariktning för I²C - för att skriva till en enhet. Satt i *i2cmaster.h*
I2C_READ	**1**	Datariktning för I²C - för att läsa från en enhet. Satt i *i2cmaster.h*
STEP_ACC_TRIGGER	**2.3**	Tröskelvärde för att bestämma om ett steg har tagits.
SET_IDLE_LOOP	**20**	Antal värden som ska hämtas från accelerometern för att bestämma “idle”-värde. Det värde som accelerometern ger när den anses ligga helt stilla.
DISPLAY_LINE_HEIGHT	**16**	Pixelhöjd för en rad på displayen. Används vid uskrift av text.
DISPLAY_MIDDLE	**(DISPLAY_LINE_HEIGHT * 2 + 2)**	Ungefärligt pixelvärde för utskrift på displayens mitt.
DISPLAY_SLEEP_DELAY	**3000**	När variabeln **displaySleepTimer** når detta värdet sätts displayen i sömnläge.

2.5.4 Funktioner

Beskrivning av egenskrivna funktioner i projektets kod

Namn	Beskrivning
double getAcc(int addr);	Hämtar accelerationsdata från MPU och ger värdet i retur. Som argument tar funktionen adressen till det register som ska läsas. Dessa adresser är definierade som macron **X, Y** och **Z.**
void drawSteps(uint16_t steps);	Skriver ut aktuellt antal steg till displayen.
void drawString(char * string);	Skriver ut en text till displayen. Texten ges som argument till funktionen, i form av en pekare till en char-sträng.
void setAccIdle();	Sätter ett värde till den globala variabeln **accIdle.** Funktionen körs endast en gång vid varje uppstart av programmet.
double getAccXYZ(void);	Använder sig av funktionen **getAcc** för att hämta accelerationsdata från **X-, Y-** och **Z**-riktningen. Det adderade värdet ges i retur.
void toggleDisplaySleep(void);	Sätter displayen i sömnläge, eller väcker displayen.

3 Referenser

Länkar till författare av kodbibliotek
MPU 6050 library - Davide Gironi
U8glib - Olikarus
I2C master library - Peter Fleury

Länkar till använda datablad för komponenter
MPU9250
Atmega328p
SSD1306 Oled Display 128x64

4 Slutsats

Skrivs senare